Practical Hardware Debugging: Quick Notes On How to Simulate Altera's Nios II Multiprocessor Systems Using Mentor Graphics' ModelSim

Ray Duran
Staff Design Specialist FAE, Altera Corporation
408-544-7937
rduran@altera.com

1. Abstract

As memory and logic in today's FPGAs has increased, multiprocessor soft cores have become a reality. Open source software tools and hardware script engines allow engineers to quickly develop software and hardware. However, debugging a multiprocessor system can be problematic. While a debugger can give very useful information about the state of a CPU in a multiprocessor environment, useful hardware information is also important. Although many tools are available to assist, most are not free. This session provides some useful PLI routines that engineers can use in conjunction with Mentor Graphics ModelSim® software and the Altera® Nios® II soft processor to help debug shared resources using MUTEX functions when simulating soft-core multiprocessor systems.

2. Altera Support Tools Summary

2.1. Nios II Embedded Processor

Altera's Nios II soft-core embedded processor is scalable for economy, small, and fast sizes. The economy size only takes about 700 logic elements (LEs) and uses only two M4K blocks, while the fast edition uses up to 1,800 LEs and three M4Ks plus cache while operating up to 51 DMIPS at a clock frequency of 50 MHz.

The Nios II processor is a general-purpose RISC processor providing:

- Full 32-bit instruction set, data path, and address space
- 32 general-purpose registers
- 32 external interrupt sources
- Dedicated instructions for computing 64-bit and 128-bit products of multiplication
- Floating-point instructions for single-precision floating-point operations
- A JTAG debug module
- Integration to a GNU C/C++ tool chain and the Eclipse Integrated Development Environment (IDE)
- An industry-standard architecture (ISA) used by all Nios II processors
- Performance up to 250 DMIPS

A Nios II processor can be built with different specifications. Software and hardware breakpoints as well as data, instruction, and off-chip traces with extra hardware are available. Other features include: bundling caches, tightly coupled memories, and custom instructions.

2.2. SOPC Builder

Altera's SOPC Builder tool helps designers rapidly develop an embedded system using a simple graphical user interface (GUI) to:

- Integrate intellectual property (IP) from Altera or Altera Megafunction Partners Program (AMPP℠) partners using the Avalon® interface, which supports logic reuse.
- Generate hardware description language (HDL) through the use of a system interconnect fabric.
- Provide a simple way of making edits and changes to a system.

The Avalon interface provides many components that are pre-built with associated drivers, including DDR memory, PCI, UARTs, serial peripheral interface (SPI), and Ethernet chips.

The system interconnect fabric uses minimal FPGA logic resources to support data path multiplexing, address decoding, wait-state generation, peripheral address alignment (including support for native or dynamic-bus sizing alignments), and interrupt priority assignments.

After building the SOPC system, an SOPC Builder system file (.ptf) that represents the hardware of the embedded system is created. The software environment uses this file as a hardware platform.

2.3. Integrated Development Environment (IDE)

The Nios II IDE is the primary software development tool for the Nios II family of embedded processors. The IDE provides project management support and compilation and flash programming, as well as a debugger with the following basic features:

- Run control
- Software breakpoints
- Disassembly viewing

In addition to targeting a logic simulator such as ModelSim software, the IDE Debugger can also connect to real hardware via JTAG and supports an ISA Nios II implementation.

3. Intent of Usage

This demo provides the user with a template (demo) and notes on how to simulate a Nios II multiprocessor system using Altera's Nios II IDE and SOPC Builder, and Mentor Graphics ModelSim simulation tool. This paper provides guidelines and information on how to simulate Avalon peripherals interacting with two or more processors.

The PLI routines attached show that useful simulation material can be recovered from the design. The design could have also been coded in HDL, C++ or other verification languages.

4. Test Scenario

A simple test scenario was chosen to illustrate that a simulation could run with two Nios II processors using ModelSim software. The entire system is shown in Figure 1.

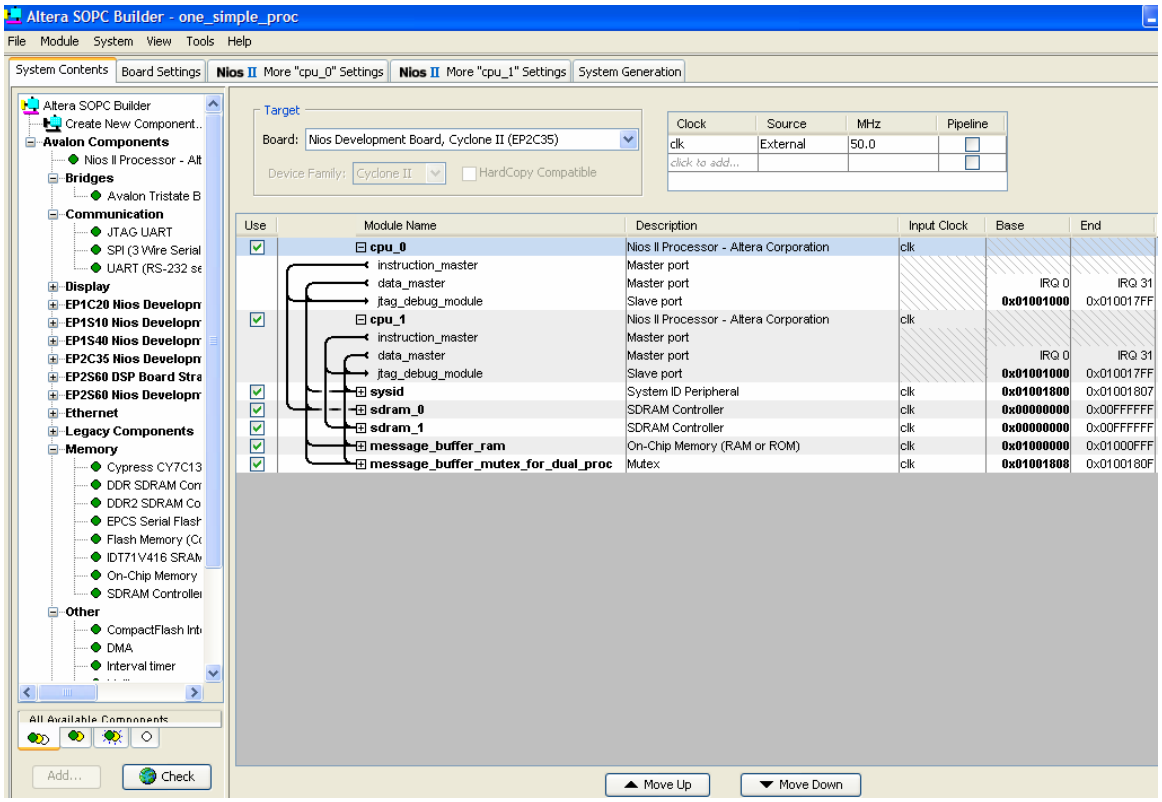

Figure 1

To simplify the demo and make debugging easier, each processor was given its own memory. To arbitrate access to the shared resource, a simple counter is used instead of a timer, which would have added more fetches for initialization.

4.1 Building Test Scenario

4.1.1. Build SOPC System

The SOPC system is built by selecting the peripherals from a menu. Each peripheral is named and can be edited for different properties. After adding all components, the user can then connect them and decide on arbitration for components.

1. Rename all components. These names will be used by software to reference the peripherals.
2. Set a reset and exception address for each processor. Use the default JTAG breakpoint assigned by the system. No interrupts are used for this demo.
3. Assign addresses to all components.
4. Accept the defaults for board and clock. *(Note: The choice of board is not important at this time because the system will only be simulated using ModelSim software.)*
5. Generate system. *(Note: The use of Reset vectors that point to volatile memory is not recommended when building real hardware, but can be used for simulation.)*

4.1.2. Build Software Project

For this demo, each processor was built in a separate project. Each project includes a number of switches that need to be set.

1. Set Small C library, clean exit (buffers), reduced drivers and ModelSim software only as switches in the properties of the project.
2. Set Program Memory, .rodata, .rwdata, heap and stack memory to the private memory of the processor.
3. Specify processor to be used.
4. Set auto-generating link script to be used.
5. Set stdout, stderr, and stdin as null for this demo.
6. Set clocktimer, and timestamp also as null.
7. Set RTOS to single thread.
8. Specify system PTF as hardware to use.

4.2. Operation of Test Scenario

Each processor runs its own code that tries to lock a MUTEX and write a value to a register. The shared resource, while only a simple register, represents a peripheral that can be developed to be shared by the two processors. In this example, each processor can only access the register only after locking the MUTEX.

4.2.1. Run ModelSim Software

After changing the directory of ModelSim to the location of the DO script setup_sim.do, compile:

1. Custom Verilog source code
2. Generated Verilog code form SOPC
3. PLI routine
4. Generated Testbench from SOPC
5. Referenced DAT file with image of each processor

If waveforms are needed, they can also be displayed from the setup window.

After simulation, the values written to the register by each processor are displayed, showing that two Nios II processors can communicate to a shared hardware resource.

5. Conclusion

The entire demo, including notes, source code and instructions are available at:

www.mentor.com/user2user

Notes and suggestions on how to simulate custom peripherals in an environment with multiple Nios II processors are also available.

6. References

[1] AN 351 Simulating Nios II Embedded Processor Designs

[2] Creating Multiprocessor Nios II Systems Tutorial

[3] The Verilog PLI Handbook Second Edition PLI Handbook, Sutherland, Stuart Sutherland HDL Inc.

[4] Nios II Software Developer's Handbook, Section II HAL System Library

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com